DESIGN AND CONSTRUCTION OF A CLOCK CONTROLLER

BY

OLAOYE MATHEW OLAYEMI

99/8309EE

DEPARTMENT OF ELECTRICAL/COMPUTER ENGINEERING

SCHOOL OF ENGINEERINGAND ENGINEERING TECHNOLOGY

FEDERAL UNIVERSITY OF TECHNOLOGY

MINNA

NOVEMBER 2005

DESIGN AND CONSTRUCTION OF A CLOCK CONTROLLER

BY

OLAOYE MATHEW OLAYEMI

99/8309EE

SUBMITTED TO

THE DEPARTMENT OF

ELECTRICAL/COMPUTER ENGINEERING

SCHOOL OF ENGINEERING AND

ENGINEERING TECHNOLOGY

FEDERAL UNIVERSITY OF TECHNOLOGY

MINNA

IN PARTIAL FUFILMENT OF THE AWARD OF

BACHELOR OF ENGINEERING (B.ENG) IN

ELECTRICAL AND COMPUTER ENGINEERING

NOVEMBER 2005

DECLARATION

I hereby declare that this project is carried out by me – OLAOYE MATHEW OLAYEMI and that the contents are a result of my own design and calculation. Information obtained from published and unpublished works of others have been well acknowledged by means of reference

OLAOYE MATHEW O.

99/8309EE

07/12/05

DATE

CERTIFICATION

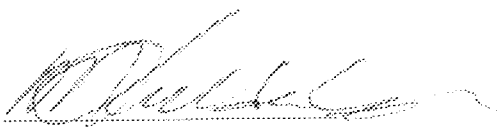
This is to certify that this thesis (Design and construction of a clock controller) is the original work of Oloye Mathew Olayemi carried out under the supervision of Mr. Nathaniel Salawu for the award of Bachelor of Engineering (B.Eng) in Electrical and Computer Engineering Department of Federal University of Technology Minna.

07/12/05

MR. NATHANIEL SALAWU

(SUPERVISOR)

DATE



08/12/05

ENGR M.D ABDULLAHI

(HEAD OF DEPARTMENT)

DATE

.....

EXTERNAL EXAMINER

.....

DATE

ACKNOWLEDGEMENT

All thank and praise to almighty God, the most gracious, the most merciful and the most supreme that has been seeing me through the course of my study.

I acknowledge with sincere gratitude, the concern and encouragement given to me by my worthy parents, Mr. and Mrs. J.O.Olaoye.

My immeasurable gratitude goes to my supervisor in person of Mr. Nathaniel Salawu, whose advice, contribution and encouragement were instrumental to the concept and completion of this project.

I am greatly indebted to my worthy lecturer, Mr. Emmanuel Eronu for his support during the design of my project.

My thanks to the following people for their immense support, both financially and physically, to the successful execution of my project, people like: Dr. A A. Oloaoye, Olaoye Felix, Akande Adeleke and Akoko Sunday.

DEDICATION

This project is dedicated to my able lecturer in person of Mr. Emmanuel Erong, whose from his immence effort I developed the programming knowledge that I used for this project

ABSTRACT

This project provides a programme timer (clock controller) designed to carry an A.C load connected to its output with a digital readout for hour and minute respectively, and can as be used as a normal clock. The main objective of this project is to use the timer for controlling the ON and OFF time (maximum of twenty-four hours) of the home appliance such as in cooker, Air – conditional and light control.

The design includes microcontroller which is programmed in C language to achieve the timing operation: four seven – segment displays for displaying the time, for control buttons for setting the time; and the relay connected to the microcontroller output. The relay is activated whenever the time set elapses and carry the AC load connected to it.

Microcontroller is used as the backbone of the design, because it requires less space, it is flexible. The operation of the timer is subjected to changes under software manipulation as compared to logic gates which cannot be altered.

TABLE OF CONTENTS

CHAPTER ONE

1.0 INTRODUCTION

1.1 SCOPE OF THE PROJECT

The project has been titled CLOCK CONTROLLER, programmable timer that is normally found in consumers' electronics such as in microwave ovens, CD players, Security systems, Attendance clocks and Digital clocks, e.t.c

The clock controller is a period timer which gives an adjustable timing period for applications such as appliance control, cooker control, AC and light control. It is a twenty-four hour clock with digital readout for the minute and hour respectively, and can be used as a normal digital clock when it is not used to control anything. The clock controller is installed in the electrical appliance, and depending on the time we want the appliance to operate, the control switches are pressed to set the appropriate time for the appliance to work.

As with other digital clock devices , the clock controller has three basic parts namely , the digital readout, which contains four seven-segment displays; microcontroller; and the output relay that is used to activate the load connected to the microcontroller.

The timing operation is carried out in software using AT98C51 microcontroller, very small microcomputer on chip use for the purpose of controlling some devices.

1.2 INTRODUCTION TO MICROCONTROLLER

A microcontroller (abbreviated MCU) is a single computer chip, integrated circuit that executes a user program for the purpose of controlling devices. The program is normally contained either in a second chip, called the EPROM – electrically programmable read only memory, or within the same chip as microcontroller itself.

A microcontroller is found in devices such as microwave ovens, automobile, keyboards, CD players, VCRs security systems e.t.c. They are used in devices that

require some amount of computing power, but don't require a much as much computing power as that provided by a complex, expensive 486 or Pentium system .They are smaller ,more reliable and cheaper .They are also ideal for the types of applications describe above where cost and unit size are very important considerations. In such applications it is almost always desirable to produce circuits that require the smallest number of integrated circuits, that require the smallest amount of physical space ,require the least amount of energy ,and cost as little as possible .

An EPROM (electrically programmable read only memory) is an integrated circuit that store program code or data, but which is maintained even when the power to the EPROM is turned off. Once the software for the microcontroller is developed, it is burnt, with special software and hardware, into the EPROM chip and the chip is physically inserted into the circuit that needs it if the program memory on the microcontroller itself is not enough.

1.3 THE SIGNIFICANT OF THE PROJECT

The significant of clock controller, programmable timer are many. Firstly, it can be used to time the operation of any home appliance or machinery. The main aim of this project is to use the controller to automatically switch off or on all office electrical equipments after some time leave the office to prevent any fire outbreak if we forget to switch them off .The timer can be set to any time to operate the load connected to it .It can also be used in time critical applications such as in rocket launching and security systems. Clock controller lets us conserve energy when used to set number of hours of the day certain electrical load connected to it to be operated , for example to automatically operate flood light for a particular time and switch it off later .

The application of clock controller can also be found in industries to automate industrial processes.

1.4 JUSTIFICATION FOR THE PROJECT

Modern electronics design is all about embedded system design. An embedded system is a system which uses the capability of microcontroller as a central control unit for the purpose of controlling the whole system and the outside world.

Microcontroller has changed the way electronics systems are been designed in recent years .This is because microcontroller, been a very small microcomputer, has a countless capabilities .It has good timer control, interrupt, great numbers of input/output and can be programmed in software.

1.5 METHODOLOGY TO BE ADOPTED IN CARRYING OUT THE PROJECT

The project uses two power sources: three 1.5V batteries to give 4.5V and secondly, the rectified AC main from the transformer .This arrangement provides a steady power supply to the microcontroller even if there is power failure.

The four seven-segment displays used are common anode, and are connected in parallel, multiplexed, to reduce the number of input/output lines used. This multiplexing is carried out in software.

The four switches are attached to the common of each seven-segment display to switch current to each segment and to set the appropriate time(hour and minute),and to manually switch on or off the clock controller .The fourth is used to save the set time .

The operation of the clock controller, microcontroller, is programmed in C language .The industry standard C51 specially developed for 8051 family of microcontroller.

1.6 LIMITATION TO THE PROJECT

The following are the main obstacle to the project:

1. COST: The cost of the 8051 programmer is very high for a student to buy. Secondly, microcontroller is very costly and scare to buy in Nigeria, making it difficult to get another when one gets damaged.

2. NON AVAILABILITY OF PROGRAMMER: There is no 8052 or 8051 programmer in the department, one need to search for them on his/her own.

CHAPTER TWO

2.0 LITERATURE REVIEW

2.1 TIMER

The idea of time is familiar to everyone, but it is a concept which is difficult to describe in scientific and philosophical terms. In science, and especially in the theory of RELATIVITY, a period of time can be treated very similarly to a distance in space[7] , hence it is loosely called the 'fourth dimension'- in addition to the three spatial dimension of length ,breath and we are simply moving along a track in four dimension ,because this would mean that the future is predestined and cannot be altered by 'free will'.

The feeling of time passing is an intuitive one, but the measurement of an interval of time requires a regularly repeating process against which it can be compared. On earth most important recurrent phenomenon is the day ,the attention of light and dark .Most species have an inbuilt 'biological clock ' which controls the animals biochemistry roughly in time with day and night , even when it is placed in an artificial non- varying environment .

The earliest clocks for measuring time during the day were SUNDIALS, where the position of shadow of a stick, or gnomon, marked the hour. Daylight was divided into a fixed number of hours, whose length varied with the time of the year .The introduction of mechanical clocks, beginning with the CLEPSYDRA, or WAKE clock, about 1500 BC, lead to the division of the day into the 24 equal hours that we use today.

The first weight driven clocks were made towards the end of the thirteenth century, but these were not very accurate because their time keeping ability depended on the function acting on a horizontal swinging beam ,and it was impossible to keep this

constant . During successive centuries many improvements were made, and the best pendulum clocks used today keep time to a thousandth of seconds per day

The most accurate clocks now available, however, are not mechanical but utilize the vibration of atoms, or of the electrons in atoms. QUARTZ clocks ,introduced in 1929[7] ,were immediately ten times more accurate than the beat pendulum clocks an have subsequently been improved . The recent development in electronic and microprocessor has made mechanical clock the thing of the past. Electronics clocks are very accurate; therefore, find application in all aspect of our life.

2.1.1 PERIOD TIMER

The most common type of timing device is undoubtedly the period timer which gives fixed or adjustable timing periods for applications such as appliance control, process timing and cooker control The timing sequence is started either manually or automatically on receiving a signal, usually electrical, from some other equipment. Some models incorporate automatic resetting for continuous recycling applications while others have to be reset by hand.

2.1.2 INTERVAL TIMER

An interval timer operates a set of contacts during a preset time interval and, at the end of the interval, returns the contacts to their normal positions This is similar in function to a time- delay relay ,except that the time interval is controlled much more accurately ,and the time interval may be many times longer than achievable with time –delay relays. Examples of the use of interval timer are in the control of exposure tie in photographic reproduction processes and the control of the duration pf a spot-welding operation and e.t c.

2.2 THE 8051 MICROCONTROLLER

The 8051 family of microcontroller was introduced in the early 1980's by Intel. Since its introduction[5], the 8051 has been one of the most popular microcontrollers and has been second sourced by many manufacturers .The 8051 currently has many different versions and some types include on-chip analogue-to-digital converters, a considerably large size of program and data memories, pulse width modulation outputs, and flash memories that can be erased and reprogrammed by electrical signals.

The 8051 family is a popular, industry standard 8-bit single chip microcontroller, manufactured by various companies with many different capabilities. The basic standard device, which is the first number if the family, is the 8051, which is a 40-pin microcontroller. This basic device is now available in several configurations. The 80c51 is the low power cmos version of the family .The 8751 contains EPROM program memory, used mainly during development work.

The 89C51,which is used for this project, contains flash programmable and erasable memory (PEROM) where the program memory can be reprogrammed without erasing the chip with ultraviolet light. At the lower end of the 8041 family we have 20-pin microcontrollers which are code compatible with the 40-pin devices .The 20-pin devices have been manufactured for complex applications where the I/O requirements are not very high and where less power is required (e.g. in portable applications.)

The AT89C1051and AT89C2051,manufactured by Intel are such microcontrollers, which are fully code compatible with the 8051 family and offer reduced power and less functionality.

The table below gives a list of the characteristics of some members of the 8051 family.

Table 1.0 some popular members of the 8051 family

Device	Program Memory	Data Memory	Timer/counter	I/O pins	Pin count
AT89C1051	1K FLASH	64 RAM	1	15	20
AT89C2051	2K FLASH	128 RAM	2	15	20
AT89C51	4K FLASH	128 RAM	2	32	40
AT89C52	8K FLASH	256 RAM	3	32	40
8051AH	4K ROM	128 RAM	2	32	40
87C51H	4K ROM	128 RAM	2	32	40
8052AH	4K ROM	256 RAM	3	32	40
87C52	8K ROM	256 RAM	3	32	40

2.2.1 ARCHITECTURE OF THE 8051 MICROCONTROLLER

The 8051 is an 8-bit, low-power, high-performance microcontroller. There are a large number of devices in the 8051 family with similar architecture and each member of the family is downward compatible with each other.

The basic 8051 microcontroller has the following features:

1. 4k bytes of program memory
2. 32 programmable I/O lines
3. 256x8 RAM of data memory
4. Six interrupt sources
5. Two 16-bit timer/counters
6. Programmable serial port
7. External memory interface
8. Standard 40-pin package

The EPROM versions of the family are used for development and the program memory of these devices id erased with an ultraviolet light source. The pin configuration of the standard 8051 microcontroller is shown in fig. 2.0 below.

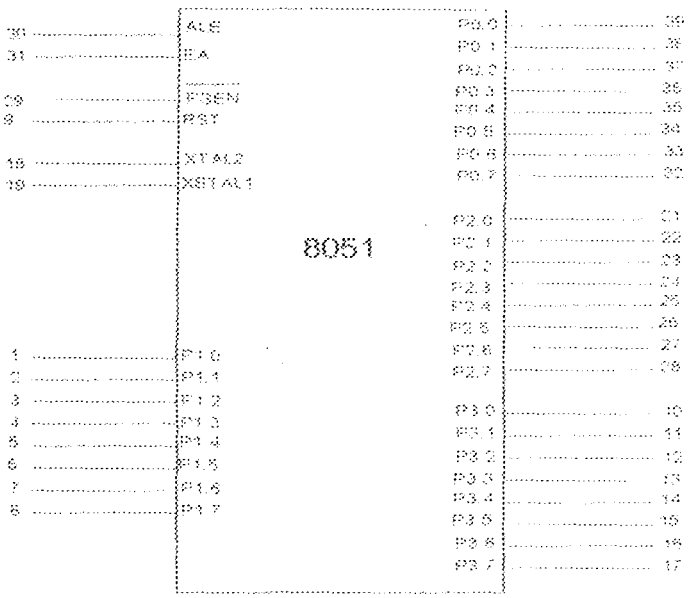


Fig.2.0 microcontroller pin configuration

DESCRIPTION OF THE PINS

RST: This is the reset input .This input should normally be at logic 0.A reset is accomplished by holding the RST pin high for at least two machine cycles power-on reset is normally performed by connecting an external capacitor and resistor to this pin.

P3.0: This is a bi-directional I/O pin with an internal pull-up resistor .The pin also acts as the data received input when the device is used as an asynchronous UART to receive serial data.

P3.1: This is a bi-directional I/O pin with an internal pull-up resistor. The pin acts as the data transmit output on the 8051 when the device is used as an asynchronous UART to transmit serial data.

XTAL1 AND XTAL2: These pins are where an external crystal should be connected for the operation of the internal oscillator. Normally two 33pf capacitors are to be connected in parallel with the crystal for operation of microcontroller.

P3.2: This is a bi-directional I/O pin with internal pull-up resistor. The pin is used as the external interrupt 0 of the microcontroller.

P3.3: This is a bi –directional I/O pin with internal pull-up resistor. The pin is used as the external interrupt 1 of the microcontroller.

P3.4: This is a bi-directional I/O pin with an internal pull-up resistor. The pin is the microcontroller counter 0 input.

P3.5: This is a bi-directional I/O pin with an internal pull-up resistor .The pin is the microcontroller counter 1 input.

GND: This pin is the ground output of the microcontroller.

P3.6: This is a bi-directional I/O pin. The pin is the external data memory read (RD).

P1.0: This is a bi-directional I/O. The pin has no internal pull-up a resistor on the 20-pin device .It is used as the positive input of the analogue comparator on the 20-pin device.

P1.1: This is a bi-directional I/O pin. It has no pull-up resistors. It is used as the negative input of analogue comparator.

P1.2 TO P1.7: These are the remaining bi-directional I/O pin of port 1.These pins have no special use ,the programmer can use the m his/her own use.

VCC: This is a power supply input pin of the microcontroller.

P0.0 TO P0.7: These are the eight I/O pins of port 0 of the standard 8051 .These pins have no pull-up resistors .They are used to provide the low addresses (A0 to A7) and the data during fetches from external program memory and during accesses to external data memory.

P2.0 TO P2.7: These are the eight I/O pins of port 2.These pins have no pull-up resistors. They are also used to provide the high addresses (A8 to a15) byte during fetches from external program memory and during accesses to external data memory.

EA/VPP: This is the external access enable pin on the standard 8051.EAshould be connected to VCC for internal program executions. This pin also receives the programming voltage during programming.

PSEN: This is the program store enable pin on the 8051. It is activated when the device is executing codes to external memory.

ALE: This is the address latch enable pin on the 8051. The pin is used to latch the low byte of the address during accesses to the external memory.

2.2.2 8051 MICROCONTROLLER TIMER

8051 microcontroller is very flexible and only choice for timing applications such as clock controller. It comes equipped with two timers, both of which may be controlled, set, read, and configured individually. The timers have three general functions namely: keeping time and/or calculating the amount of time between events; counting the events themselves; and generating baud rates for the serial port.

The timer always counts up. Obviously, one of the primary uses of timers is to measure time. these timer/counters can be operated in several different modes depending upon the programming of the two registers RCON and TMOD as shown in the table 1.1 and 1.2 below. These registers are programmed before using them.

Table 2.0 timer/counter control registers (TCON)

Bit Name	Bit position	Description
TF1	7	Timer 1 overflows flag. Set and clear by hardware
TR1	6	Timer 1 run control bit. Timer 1 timer is turned on when TR1=1
TF0	5	Timer 0 overflows flag. Set and clear by hardware
TR0	4	Timer 0 run control bit. Timer 1 timer is turned on when TR0=1

Table 2.1 timer/counter mode control register (TMOD)

TIMER 1				TIMER 0			
GATE 1	C/T 1	T1M1	T1M0	GATE 0	C/T 0	T0M1	T0M0

GATE: When TRx is set and GATE=1, timer runs only while the intx pin is high .If GATE=0 timer will run only while TR_X =1

C/T: Timer/ counter select bit When C/T=0, it operates as a timer. When C/T=1, it operates as a counter.

M1, M0: These bits select the mode of operation

CHAPTER THREE

3.0 HARDWARE DESIGN

3.1 AT89C51 MINIMUM REQUIREMENT

For AT89C51 microcontroller to work properly, the manufacture has specified the components to connect to the external pins before it can be used.

The configuration as shown in the figure 3.1. Starting from the power supply to RESET pin. The 10μf capacitor and 8.2kΩ resistor connected in parallel to the RESET pin let the microcontroller automatically reset it when it is boot – up.

Secondly, the external crystal connected across the X1and X2, and with two 33pf capacitor in parallel wit it, set the microcontroller clock and lastly, the GND pin stands for the ground.

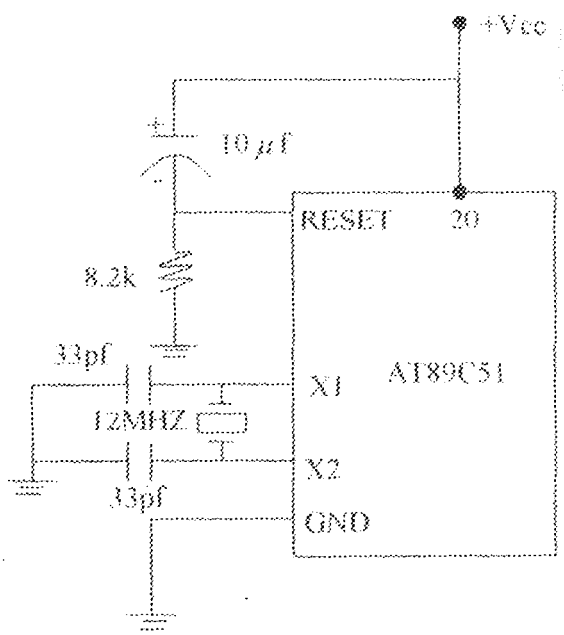


Fig 3.1 AT89C51 MINIMUM REQUIREMENT

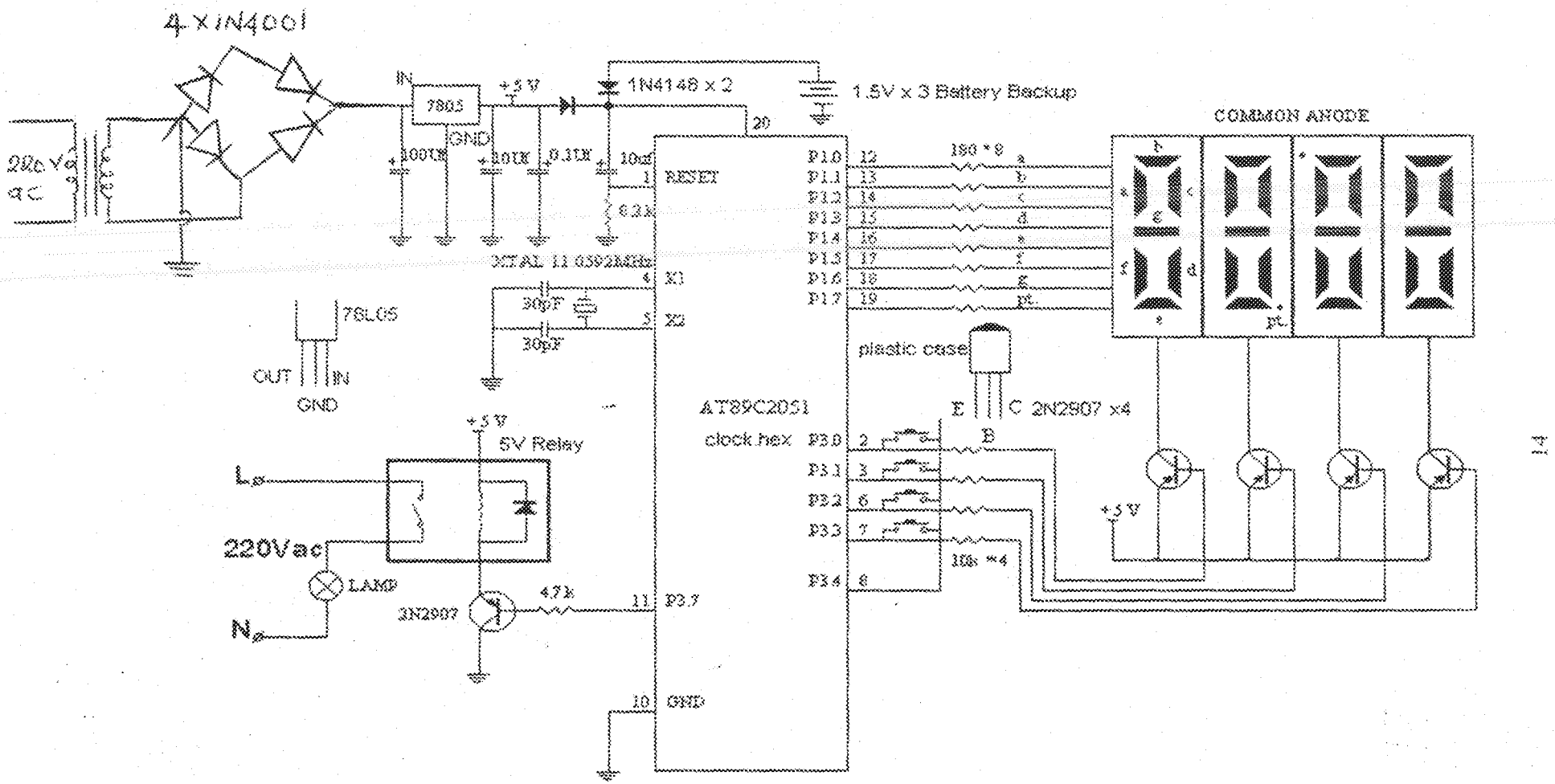


FIG.3.1 THE CIRCUIT DIAGRAM OF A CLOCK CONTROLLER

3.2 TIMER DISPLAY AND CONTROL UNIT

To easily understand the analyses and calculation to be carried out, the complete circuit diagram of the clock controller is shown in figure 3.2

The output display is an interconnection of four 7 – segment LEDs in parallel, and by this arrangement we say they are multiplexed. Using the same parallel lines to supply current to all the segments and rising transistor to switch them on one after the other

These 7-segments are common anode which requires logic 0 to turn on particular segment to form the decimal digit. Common anode makes provision for external power supply to be used instead of the microcontroller sourcing the total current itself, and as a result of that may damage it.

PNP transistor is used as a switch to control and supply current to each segment respectively for a particular digit to be displayed, all other digit have to be switched off. This is controlled in software in very short time, switching between the segments, that human eyes cannot notice, and also control can be performed pressing the four button connected to the base of the transistor.

Let us calculate the switching current of each transistor connected to the common anode of each digit. If one segment is used to represent all the seven segment, as shown below.

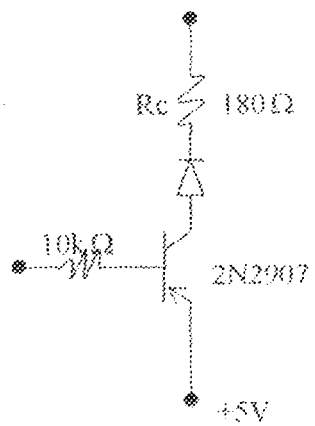


Fig 3.3 LED driver

Since positive power supply is used throughout in the circuit, [1] the connection of the PNP transistor is turned upside down as compared to the negative supply.

To calculate the saturation current, PC (sat), and cut – off voltage of the transistor

Let $I_B = \frac{V_{BB}}{R_B}$

$= \frac{4.5V}{10 \times 10^3} = 0.45mA$

Calculating the current one segment:

Let $I_S = \frac{V_{CC} - V_D}{R_S}$

WHERE I_S = supply current

V_{cc} = supply voltage

V_D = voltage drop across segments

$\therefore I_S = \frac{5 - 1.8}{180} = 17.78mA$

$\therefore I_C (sat) = I_S \times 7$

$= 17.78 \times 10^{-3} \times 7$

$= 124.46$ mA

Now, at cut-off, $I_C = 0$, therefore:

$I_C = \frac{V_{cc} - V_{CE}}{R_C}$

$0 = \frac{V_{CC} - V_{CE}}{R_C}$

$0 = \frac{3.2 - V_{CE}}{180}$

V_{CE} (cut- off) = 3.2V

With transistor current gain of 300, the collected current will be.

$I_C = \beta ac \times I_B$

$= 300 \times 0.45 \times 10^{-3}$

135mA

Obviously, I_C cannot be that large because its maximum value is given by $(V_{CC} - V_B)/R_B = 3.2/180 = 17.78$mA however, let's assume that I_C takes this value temporarily. Then:

$V_{CE} = V_{CC} - Ic \times RL$

$= 3V - 300V$

$= -10.3V$

This means that the transistor is working well into saturation.

3.3 THE OUTPUT RELAY

This section of the project is the output to which the load is connected. Transistor is used to switch the very small current from pin P3.7 of port 3 to the relay, and as a result of this, very large current can be switched into any load connected to the relay terminals.

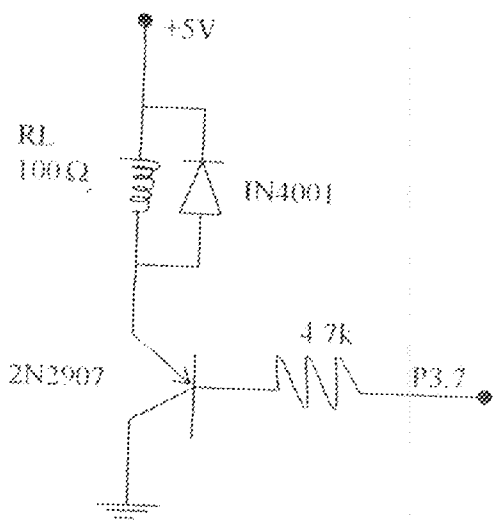


FIG 3.4 Relay driver

At saturation, $V_{CE} = 0$

$$I_C = \frac{V_{CC} - V_{CE}}{R_L}$$

$$= \frac{5 - 0}{100} = 50mA$$

At cut – off, $I_C = 0$

Therefore

$$I_C = \frac{V_{CC} - V_{CE}}{100}$$

$$0 = \frac{5 - V_{CE}}{100}$$

$V_{CE} = 5V$

To calculate the base current, I_B

$$I_B = \frac{V_{BB}}{R_B} = \frac{4.5}{4.7 \times 10^3}$$

$= 0.96mA$

To know whether the transistor is working well into saturation

$I_C = \beta_{DC} \times I_B$

$= 300 \times 0.96 \times 10^{-3}$

$= 0.29A$

$V_{CE} = V_{CC} - I_C R_L$

$= 5 - 0.29 \times 100$

$= -24V$

The transistor is very well into saturation.

3.4 POWER SUPPLY UNIT

The current requirement of AT89c51 is 20mA. Taking into account the saturation current, I_L(sat), calculated from the previous calculation. Therefore, the total current required is:

I_T = (124.46 + 20+50) mA

= 194.46mA

With full-wave bridge rectifier, the frequency of rectification is twice that of half-wave rectifier, therefore the frequency is multiplied by two. Since supply frequency is 50HZ, the full-wave frequency is 100HZ.

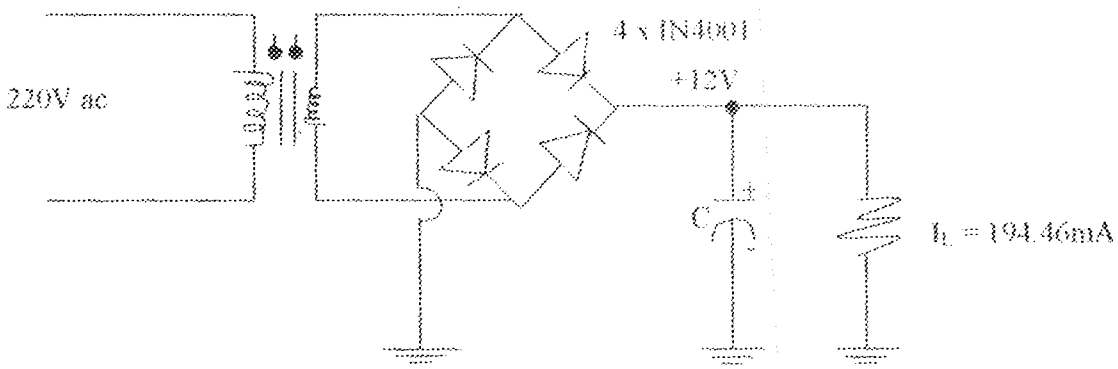


Fig 3.5 Rectified power supply

Assume total peak-to-peak voltage of less than 1V is required. For example 0.85V.

$$V_R = \frac{IL}{FC}$$

$$C = \frac{I_L}{fV_R}$$

$$= \frac{194.46 \times 10^{-3}}{(100HZ) \times (0.85V)}$$

$= 2,287.76 \times 10^{-6}$ f

The close value to the answer gotten is 2200μf

Therefore, C = 2200μf.

To better improve or reduce the output ripple of the supply voltage, an IC regulator is connected across the capacitor. LM7805 is used because of its high common mode rejection ratio of 80dB

$\therefore$ Ripple rejection, RR = antilog $\frac{80dB}{20} = 10{,}000$

Therefore, the peak-to-peak output ripple of the regulator is approximately:

$$V_n(out) = \frac{V_n}{RR}$$

$$= \frac{0.85V}{10{,}000} = 85\mu V$$

This means that LM7805 can reduce input ripple voltage by a factor of 10,000, and this makes it very good digital circuit

The complete regulated power supply unit is shown in the figure below

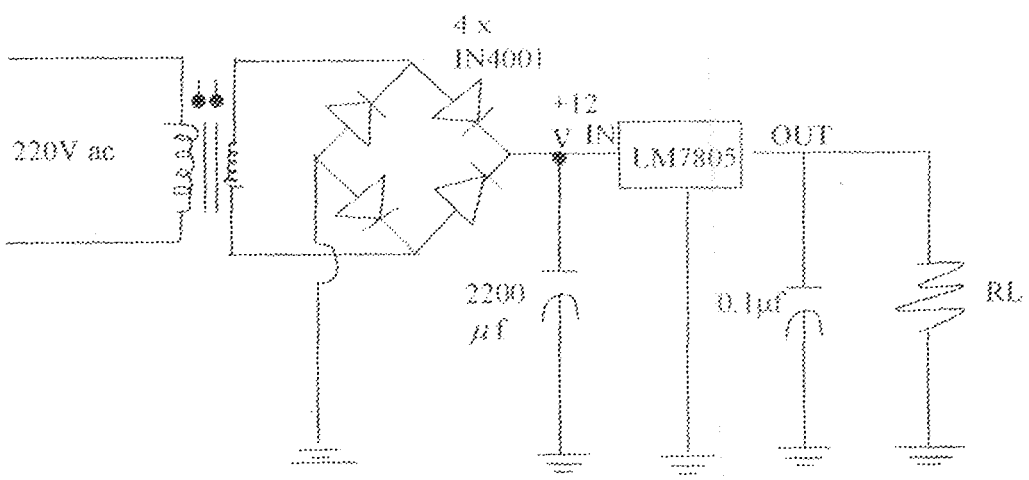


Fig 3.6 Regulated power supply

CHAPTER FOUR

4.0 SOFTWARE DESIGN

The C programming language is a general purpose high-level programming language that offers efficient and compact code and provides elements of structured programming, as compared to low-level assembly language. Many control and monitoring - based application can be solved better and efficiently with C language than with any other programming language

C was originally available on mainframe computers, minicomputers, and personal computer (PCs). The C programming language is now available on most microcontrollers and microprocessors, example of which is AT89C51 used in this project.

4.1 GENERATING SEGMENTS BIT PATTERN

By programming, or software control, the individual segment connected to port 1, as shown in fig 3.2 of chapter three, can be switch on or off. Thereby, forming the decimal number 0 to 9. This bit pattern is shown in the table 4.1 below. Because common – anode, which requires logic 0 to turn on a segment, is used. The Hexadecimal equivalent is first complimented in software before being sent out of Port 1. for example, to form digit 0, all the segment except g are turn on As a result of this, logic 1 is used to represent the ON and logic 0 for off respectively

Table 4.1 segments and bit pattern

	x	g	f	e	d	c	b	a	HEX
0	0	0	1	1	1	1	1	1	3F
1	0	0	0	0	1	1	1	0	OC
2	0	1	1	1	0	1	1	0	F6
3	0	1	0	1	1	1	1	0	5E
4	0	1	0	0	1	1	0	1	4D
5	0	1	0	1	1	0	1	1	5B
6	0	1	1	1	1	0	1	1	7B
7	0	1	0	0	1	1	1	0	0E
8	0	1	1	1	1	1	1	1	7F
9	0	1	0	1	1	1	1	1	5F

These bit patterns is then placed into one dimension array to minimize the number of program code required. The name of this array is called 'convert' and it is placed in a loop for referencing each of its elements:

Char code convert [10] = {0 × 3F, 0 × 0C, 0 × 76, 0 × 5E, 0 × 4D, 0 × 5B, 0 × 7B, 0 × 0E, 0 × 7F, 0 × 5F }.

4.2 TIMER INITIALIZATION

The most important aspect of this project is the setting of the microcontroller timer to generate precisely 10msec delay. To achieve this, let us start from the external crystal oscillation with value of 12MHZ.

With 8051 microcontroller, one instruction cycle = 12 clock cycles

Therefore, number of instruction per second = $\frac{12MHZ}{12}$

= 1,000,000 inst/sec

The timer is 16 bit and incremented per instruction cycle, therefore to make the timer overflow at a particular point that will be equivalent to 10msec.

Let X = value at which the timer overflow when it is 10msec.

$$\frac{1{,}000{,}000 inst./\sec}{1\sec} = \frac{X}{0.01\sec}$$

X = 1,000,000 × 0.01

= 10,000

To get the timer value, this value, X, is subtracted from the maximum value of timer as thus:

Timer value = 65536 – 10,000 = 55536 = D8F0.

This value is converted into hexadecimal and use to set the timer. F0 is used to set timer low byte, while D8 is used to set timer high byte as will be seen in the C program.

By this, 10msec delay has been achieved fro the program to be running, and the second thing is to get 1 sec delay and its multiple to get clock minute and hour respectively. One function in C program that does tat is Void time ( )

4.3 THE CLOCK CONTROLLER PROGRAM

```
#include <at89x51.h>
#include <stdio.h>

unsigned  char sec100,sec,sec5,min,hour,flag1,command,temp,opto;
unsigned char i,digit,buffer[4],onHour1,onMin1,offHour1,offMin1;
char cputick,key,delay,count1;

char code convert[10] = {0x3F,0x0c,0x76,0x5e,0x4d,0x5b,0x7b,0x0e,0x7f,0x5f};


void pause(int);
void scanLED();
void manualOnOff();
void savetimeOnOff1();
void setmin();
void sethour();
void showOnce();
void savetimeOff1();

void time();
void timeToBuffer();
void blink();
```

```
void offmsd();
void keyexe();
void keydelay();
void comparetime();

void timer_isr (void)  interrupt 1  using 1  {
        TH0 |= 0xdc;  // reload timer 0 with 0DC00H
        cputick++;
        time();       // update realtime clock

}

void main()
{
 EA = 1;
 ET0 = 1;  // or IE |= 0x82;   /* set bit EA and Timer0 enable */
 TMOD |= 0x01; /* timer 0 run 16 bit counter */
 TR0 = 1; //or TCON |= 0x10; /* run timer 0 */
 opto = 0xff;
 cputick = 0;
 hour = 18;
 min = 0;
 sec = 0;
 key = -1;
 flag1 = 0;
 onHour1 = 18;   /* 18:30 turn lamp on */
 onMin1 = 01;
```

```
  offHour1 = 18;   /* 21:30 turn off */
  offMin1 = 02;
  count1 = 0;
  buffer[0] = 0x40;
  buffer[1] = 0x40;
  buffer[2] = 0x40;
  buffer[3] = 0x40;

  while(1)
  {
   while ( cputick < 1)
   scanLED();

   cputick = 0;

/*-------------- the following tasks execute every 10ms -------*/

  // time();
  timeToBuffer();
  blink();
  offinsd();
  keyexe();
  keydelay();
  comparetime();

/*-------------------------------------------------------------*/
  }
```

```
    }

    /* ******************* change constant below for another X-tal ********/
    void time ()
    /* update real-time clock  */
    {
     sec100++;
     if (sec100 >= 100)      /* 100 * 10 ms = 1 s */
     {sec100 = 0;
     flag1 |= 0x05;   /* set bit 0, bit 2 */
     temp = 50;
     sec++;
     if (sec >= 60)
     {sec = 0;
     flag1 |= 0x02; /* set bit 1 */
     min++;
     if (min >= 60)
     {min = 0;
    hour++;
    if (hour >= 24)
    {hour = 0;


   }
  }
 }
}
}
```

```
void scanLED() /* scan 4-digit LED and 4-key switch, if key pressed key = 0-3
else key = -1 */

{
  int i;
  digit = 0x08;
  key = -1;
  for( i = 0; i < 4; i++)  /* 4-DIGIT scanning */
  {
    P3 = ~digit & opto;  /* send complement[digit] */
    P1 = ~buffer[i];  /* send complement[segment] */
    pause(5);         /* delay a while */
    P1 = 0xff;        /* off LED */
    if ((P3 & 0x10) == 0) /* if key pressed P3.4 became low */
      key = i;        /* save key position to key variable */
    digit>>=1;        /* next digit */
  }
}


void timeToBuffer()
{
  buffer[0] = convert[min%10];
  buffer[1] = convert[min/10];
  buffer[2] = convert[hour%10];
  buffer[3] = convert[hour/10];
```

```
    }

  void blink()
  {
    if(flag1 & 0x04) != 0) /* check bit 2 if set decrement temp until zero */
     {temp--;
       if (temp != 0)
             {
             buffer[1] |= 0x80;
             buffer[2] |= 0x80;
             }
            else( flag1 &= ~0x04);
     }
}


void keyexe()
{
  if (key != -1)
  {
    if ((flag1 & 0x80) == 0)  /* within 0.5 sec after 1st press
                        the following execution is not allowed */
      {
        flag1 |= 0x80;
        delay = 50;
```

```
        switch(key){
         case (0): /* key position 0 */
         manualOnOff();    /* service key 0 */
         break;
         case (1): /* key position 1 */
         savetimeOnOff1();  /* service key 1 */
         break;
         case (2): /* key position 2 */
         setmin(); /* service key 2 */
         break;
         case (3): /* key position 3 */
         sethour();
                }
               }


    }
 }

 void sethour()
 {
   hour++;
   if ( hour== 24)
   hour = 0;

}

void setmin()
```

```
    {
      min++;
      sec = 0;
      if( min == 60 )
      min = 0;
    }

  void savetimeOnOff1()

  {
    count1++;
    if (count1 == 1)
    {
   onHour1 = hour;
   onMin1 = min;
   buffer[0] = 0x00;
   buffer[1] = 0x68;
   buffer[2] = 0x78;
   buffer[3] = 0x71;
   showOnce();
   }
   else
   {
     count1 = 0;
     savetimeOff1();
   }
}
```

```
void savetimeOff1()

{
  offHour1 = hour;
  offMin1 = min;
  buffer[0] = 0x63;
  buffer[1] = 0x63;
  buffer[2] = 0x78;
  buffer[3] = 0x71;
  showOnce();
}

void manualOnOff()

{
  opto = ~opto | 0x7f;  /* complement bit 7 which in turn activates P3.7 */
  if ((opto & 0x80) == 0)
  {
    buffer[0] = 0;
    buffer[1] = 0;
    buffer[2] = 0x68;
    buffer[3] = 0x78;
    showOnce();
  }
  else
  {
```

```
            buffer[0] = 0;
            buffer[1] = 0x63;
            buffer[2] = 0x63;
            buffer[3] = 0x78;
            showOnce();
        }
    }

    void showOnce()
    {
      int i;
      for(i=0;i<2000;i++)
      scanLED();
    }
    void keydelay()
    {
      if ((flag1 & 0x80) !=0)
          {
            delay--;
          if(delay == 0)
            flag1 &= ~0x80;
          }
}

void comparetime()
{
  if((flag1 & 0x01) != 0 )
```

```
    {
      flag1 &= ~0x01;
    if(hour == onHour1 && min == onMin1)
        opto = 0x7f; /* clear P3.7 turning opto on  */
    if(hour == offHour1 && min == offMin1)
        opto = 0xff; /* set bit P3.7 turning opto off */
    }
}

void offmsd()

{
  if (buffer[3] == 0x3f)   /* if msd = '0' then put blank unstead */
    buffer[3] = 0x00;
}


void pause(int j)
{
  int i;
  for (i = 0; i < j; i++)
  ;
}
```

4.3.1 PROGRAM DEFINITION

To briefly explain the operation of the program, let us look at the import parts (functions) of the program.

1. void keyexe ( ): This function contains many other functions. It prevents us from pressing more than one key at time, and execute instruction for the four keys:

(i) Pressing key position 0 which is used to manually switch off or on the clock controller.

(ii) Pressing key position 1 which is used to save the ON and OFF time respectively.

(iii) Pressing key position 2 which is used to set minute.

(iv) Pressing key position 3 to set the hour.

2. void time( ): This function updates the timer and it is used to set minutes and hour overflow.

3. void offmsd( ): This function makes sure that when the most significant bit of hour is zero, it then put blank instead i.e. zero blanking.

4. Compare time( ) : This function compare the set ON and OFF times of the clock controller with the current time of the clock, when the current time is the with the set time, the output of port 3 (p3.7) is deactivated and in turn activate the relay.

5. void ScanLED( ): This function scans the 4 – digit and 4 – key switch, if key pressed.

4.4 BURNING THE PROGRAM INTO MICRO CONTROLLER

After the C program has been written and debugged, it was transferred into crimson editor which was specifically developed for editing 8051 family of microcontroller. The program was error free in crimson editor.

Secondly, from the crimson editor, the program was also transferred into SDCC (Small device C compiler) to compile the source code into hex file needed by the microcontroller. Since the source code complied without error, it showed that there was no problem with the hex file.

Thirdly, the 8051 programmer was then connected to the PC serial port and the targeted chip, AT89c51, was inserted into the socket on the programmer. The compiled hex file was transferred, from SDCC, into the chip and later verified this on the PC.

Lastly, the chip was removed and inserted into the 40-pin dual-in-line socket on the clock controller circuit.

4.5 DISCURSION OF RESULT

After the microcontroller was inserted into the circuit, it was tested and proved to be functioning. When the clock controller was first booted, there was no operation of the controller until a key is pressed. The operation of the controller depends on the pressing of the four control switches. To understand the operation of this project better, the function of each key is stated below.

1. Key 0: When this key is pressed the clock controller is manually switch on, and pressing it again will switch it off.
2. Key 1: This key is used to save the time set for the appliance or the load connected to the output of the controller to operate, when it is pressed once, it will save the on time and when it is pressed twice it will save the off time.
3. Key 2. This is used to set the minute for the appliance connected to the clock controller to operate.
4. Key 3: This is used to set the hour for the appliance connected to the clock controller to operate.

The time set is compared, every 10 ms, with the real-time clock, when these two times are the same port 3(p3.7) is deactivated which in turn activate the relay and operate the AC load connected to the output of the clock controller. The time of operation can be set to maximum of twenty-four hours.

CHAPTER FIVE

5.0 CONCLUSION AND RECOMMENDATION

5.1 CONCLUSION

After careful design of the software, it was tested with Topview simulator and proved to be successful .The software was burnt into the microcontroller and inserted into the circuit.

Some problems were encountered during the design and construction which id the unavailability of microcontroller programmer .Secondly, microcontroller itself is very scarce and costly in Nigeria.

Despite all the problems encountered, the aim of the project was achieved.

5.2 RECOMMENDATION

The scope of this work leaves much room for greater improvement, mostly due to time constraint and limited resources ,The project could be further enhanced to cater for a lot of other needs in time critical applications

The development of any country depends on her engineers ,mostly electrical engineers .The department should sensitize the students on the usefulness of programmable logic devices in modern electrical design. Countries like Japan USA, UK, Germany, e.t.c. use programmable logic in all the machines and consumer electronics imported into Africa. if this technology is transferred into many Nigerian students they will be qualify enough to meet the challenges in modern electronic design As a result of this the country will improve technologically

REFERENCES

[1] Albert Paul Malvino, Electronic principles,6th Ed., Tata McGraw-Hill publishing company Ltd.New Delhi,1999,pp.165-170,239-242,273-274,926-933.

[2] B L. Theraja and A K Theraja,Electrical technology ,21st Ed.,S chand and company Ltd .Ram Nagar,New delhi,2000.pp. 1761,1770-1784.

[3] Donald P. Leach, Albert Malvino, Digital principles and applications, 4th Ed , McGraw-Hill book company, 1986, pp.4, 5, 6.

[4] Charles A. Holt, Electronic circuits: Digital and Analog, John Wiley and sons, Inc., 1978, pp.158, 210.

[5] Dorgan Ibrahim, microcontroller projects in C for the 8951, Butterworth –Heinemann, oxford, London, 1999, pp.1-200

[6] Hintz, K. J, merging C and assembly language in microcontroller applications, Journal of microcontroller applications, Vol.15,no 1,

July, pp.267-278.

[7] How it works, encyclopaedia of science and invention, H S Stuttman publishers, New york, 1974, pp 2433-2436.

[8] www.atmel.com

[9] www.8052.com/tut8052.phtml

[10] www.microcontrollers.com

[11] www.vaultlibs.com/asm8052

[12] www.ashling.com

[13] http://sdcc.sourceforge.net/#platforms

[14] www.domtronics.com/cvsr.htm

[15] www.goecities/dincerayda/8051/51links.html